University of Southern Denmark

DM852

Heuristics & Approximations Algorithms

Submitted To:
Marco Chiarandini
Lene Monrad Favrholdt
IMADA
Mathematics & Computer Science Department

Submitted By :
Alexander Lerche Falk
Kicito Narongrit Unwe
Spring - Master of Computer Science

Contents

1 Introduction

This project shows heuristics algorithms, which can resolve the Capacitated Vehicle Routing Problem (CVRP). If the heuristic can resolve the problem, we are going to apply Local Search algorithms to improve the results from the heuristics. The CVRP gives insight into how to deliver orders to customers using the shortest route for each vehicle, while preserving the capacity limit of each vehicle.
newline We have created two heuristic algorithms: one using a nearest-neighbour terminology and the other one taking inspiration of clustering close customers. We also show two self-implemented Local Search algorithms. Lastly, we are going to compare our own ideas against more common used algorithms, which are tested and proved to work.

— Alexander & Kicito

2 Custom Heuristics Algorithms

The first algorithm, which we have created, takes the approach of the Nearest Neighbour idea. Given an instance of CVRP, the starting point the depot where the vehicles are being loaded with customer requests is the point, where all calculations start. We start by checking the nearest point from the depot and add it to the first route. We then add the shortest path from the newly added point to the route and continue until we are out of capacity. Each point (customer) has a capacity, which adds up until the max is reached. We continue during the same from adding the first shortest route from the depot until we have reached our capacity limit. At one point, we have x amounts of routes, covering the problem instance.

Algorithm 1 Custom CVRP Heuristic - Nearest Neighbour Approach

Require: $Point_1 \dots Point_N$
Ensure: *Solution* (solution to the CVRP instance)

```
1:  function ALGORITHM(Points[ ])
2:      capacity ← 0
3:      data ← CVRP instance
4:      visited ← empty array containing visited nodes
5:      shortestdistance ← 0
6:      current ← keep track of next node to visit
7:      for i ← 0 to length of data-1 do
8:          for j ← 1 to length of data do
9:              temp ← euclideandistance(current, j)
10:             if j not in visited then
11:                 if temp ≤ shortestdistance then
12:                     shortestdistance ← temp
13:                     current ← j
14:                     visited ← j
    ▷ checking if capacity requirements are met
15:                     if current total capacity ≤ max capacity then
16:                         capacity ← capacity of current node
17:                     else
18:                         Add starting point to end of route and
19:                         add capacity of current node to variable(capacity)
20:         shortestdistance ← 0                          ▷ reset shortestdistance
21:     return solution
```

The first suggestion for an algorithm has a time complexity of $O(n^2)$. The algorithm has to iterate through the points of the instance twice. Given a point in the solution

space, we are trying to find the shortest distance to any point from the current point. When we have found it, we are marking the point, and making it the new current point. Then we continue to find the shortest distance to any non-visited point, while preserving the maximum capacity of the vehicle. If we find any shortest distance point from a current point, but it is going to break the maximum capacity limit, then we are returning to the depot, and start a new route.

Our next algorithm is inspired by Cluster Analysis [1], which is popular in machine learning and data mining. The idea is to group a set of points lying close to each other in an euclidean space. By choosing such approach to attack the Vehicle Routing Problem, we can ensure points are lying close to each other before we are deriving a route.
The heuristic algorithm has the following behaviour: (1) Pick a point furthest from away from the depot, which is not in the solution route; (2) find nearest points until the capacity is reached and save the points as a cluster; (3) send a vehicle to all points in each cluster; (4) and repeat.

Algorithm 2 Custom CVRP Heuristic - Cluster Analysis Approach

Require: $Point_1 \dots Point_N$
Ensure: *Solution* (solution to the CVRP instance)
1: **function** ALGORITHM(*Points*[ ])
2: `capacity` ← 0
3: `data` ← CVRP instance
4: `visited` ← *empty array containing visited nodes*
5: `cost` ← *0*
6: **while** `i` ← 0 `to length of data-1` **do**
7: **return** *solution*

3 Custom Local Search Algorithms

While having created the heuristics, which provides us with solutions to the initial problem instances, we can still optimize. This is where the Local Search Optimization algorithms comes into play. Given a canonical solution, we want to optimize it to reduce the total cost of the solution. We have provided three custom Local Search (LS) algorithms, trying to optimize the canonical solution.
The first LS algorithm looks into the generated routes done by the heuristics. It takes two routes and compare them with each other, checking whether any swaps are possible in the two routes. A swap is possible if the distance is being reduced in at least one route, while still preserving the maximum capacity limit of both routes.

The second LS algorithm looks like the first but with a minor change. It takes one route and compares it with every other route to find a better solution. It then continues doing this with every other route; take one route, compare it with every other route.

References

[1] Wikipedia. Cluster analysis, March 2019.